Использование PYTHON для решения заданий КЕГЭ

Задание 26

Задание 26. Обработка массива целых чисел из файла. Сортировка.

- Чтение данных в список

```
data = [0]*N
with open("26.txt") as Fin:
  for i in range(N):
      data[i] = int(Fin.readline())
```

- Сортировка списка по возрастанию
- Сортировка списка по убыванию

Задание 26. Обработка массива целых чисел из файла. Сортировка.

- Чтение данных в список

```
data = [0]*N
with open("26.txt") as Fin:
  for i in range(N):
      data[i] = int(Fin.readline())
```

```
with open("26.txt") as F:
  data = [int(F.readline())
          for i in range(N)]
```

```python
with open("26.txt") as F:
  data = F.readlines()
S, N = map(int, data[0].split())
del data[0]
data = sorted(list(map(int, data)))
```

```python
data=[]
F = open("26.txt")
S, N = map(int, F.readline().split())
for I in range(N):
    a = int(F.readline())
    data.append(a)
data.sort()
```

- Сортировка списка по возрастанию

```
data.sort()
```

- Сортировка списка по убыванию

```
data.sort(reverse = True)
```

- Сортировка списка по последней цифре числа

```
def lastDigit(n):
    return n % 10
data.sort(key = lastDigit)


data.sort( key = lambda x: x % 10 )
```

Системный администратор раз в неделю создаёт архив пользовательских файлов. Однако объём диска, куда он помещает архив, может быть меньше, чем суммарный объём архивируемых файлов. Известно, какой объём занимает файл каждого пользователя. По заданной информации об объёме файлов пользователей и свободном объёме на архивном диске определите максимальное число пользователей, чьи файлы можно сохранить в архиве, а также максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

***Входные данные**. В первой строке входного файла* **`26.txt`** *находятся два числа: S – размер свободного места на диске (натуральное число, не превышающее 100 000) и N – количество пользователей (натуральное число, не превышающее 10000). В следующих N строках находятся значения объёмов файлов каждого пользователя (все числа натуральные, не превышающие 100), каждое в отдельной строке. Запишите в ответе два числа: сначала наибольшее число пользователей, чьи файлы могут быть помещены в архив, затем максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.*

Пример входного файла:

```
100 4
80
30
50
40
```

При таких исходных данных можно сохранить файлы максимум двух пользователей. Возможные объёмы этих двух файлов 30 и 40, 30 и 50 или 40 и 50. Наибольший объём файла из перечисленных пар – 50, поэтому ответ для приведённого примера:

```
2 50
```

```python
with open("26.txt") as Fin:
  data = Fin.readlines()
S, _ = map( int, data[0].split() )
del data[0]
data = sorted( list(map(int, data)) )
total = 0
for i, val in enumerate(data):
  if total + val > S: break
  total += val
print(i)
delta = S - total
candidates = [x for x in data
                if x-data[i-1] <= delta]
print( max(candidates) )
```

Организация купила для своих сотрудников все места в нескольких подряд идущих рядах на концертной площадке. Известно, какие места уже распределены между сотрудниками. Найдите ряд с наибольшим номером, в котором есть два соседних места, таких что слева и справа от них в том же ряду места уже распределены (заняты). Гарантируется, что есть хотя бы один ряд, удовлетворяющий условию.

Входные данные представлены в файле **26-59.txt** следующим образом. В первой строке входного файла находится одно число: N – количество занятых мест (натуральное число, не превышающее 10 000). В следующих *N* строках находятся пары чисел: ряд и место выкупленного билета, не превышающие 100 000.

В ответе запишите два целых числа: номер ряда и наименьший номер места из найденных в этом ряду подходящих пар.

Пример входного файла:

```
10
5 5
5 9
5 6
16 9
16 3
16 6
20 23
20 28
20 35
20 40
```

В данном примере есть следующие свободные места, удовлетворяющие условию: 7 и 8 в ряду 5, 4 и 5 в ряду 16, а также 7 и 8 в ряду 16. Выбираем наибольший номер ряда: 16 и наименьший номер места: 4. В ответе нужно указать: 16 4.

```python
data=[]
F=open("26-59.txt")
N = int(F.readline())
for i in range(N):
    z1,z2=map(int,F.readline().split())
    data.append([z1, z2])
data.sort(key=lambda x: x[1])
data.sort(key=lambda x: x[0], reverse=True)
for i in range(N-1):
    if data[i][0]==data[i+1][0] and \
       data[i+1][1]-data[i][1]==3:
        row=data[i][0]
        col=data[i][1]+1
        break
print(row, col)
```

На закупку товаров типов A, B, C, D и E выделена определённая сумма денег. Эти товары есть в продаже по различной цене. Необходимо на выделенную сумму закупить как можно больше товаров пяти типов (по общему количеству). Если можно разными способами купить максимальное количество пяти типов товаров, то нужно выбрать способ, при котором будет закуплено как можно больше товаров типа B. Если при этих условиях есть несколько способов закупки, нужно потратить как можно меньше денег.

Определите, сколько будет закуплено товаров типа B и сколько денег останется.

Входные данные представлены в файле **`26-64.txt`** следующим образом. Первая строка входного файла содержит два целых числа: N – общее количество товаров и M – сумма выделенных на закупку денег (в рублях). Каждая из следующих N строк содержит целое число (цена товара в рублях) и символ (латинская буква), определяющий тип товара. Все данные в строках входного файла отделены одним пробелом.

Запишите в ответе два числа: сначала количество закупленных товаров типа B, затем оставшуюся неиспользованной сумму денег.

Пример входного файла:

```
6 110
40 E
50 A
50 B
30 C
20 B
10 A
```

В данном случае можно купить не более четырёх товаров, из них не более двух товаров типа B.

Минимальная цена такой покупки 100 рублей (покупаем товары 10 A, 20 B, 30 C, 50 B). Останется 0 рублей.

Ответ: 2 0.

```python
data=[]
dataB=[]
dataACD=[]
F=open("26-64.txt")
N, S= map(int,F.readline().split())
for i in range(N):
    z1,z2=map(str,F.readline().split())
    data.append([int(z1), z2])
data.sort(key=lambda x: x[0])
rez=0
countB=0
for i in range(N):
    if rez+data[i][0]<=S:
        rez+=data[i][0]
        if data[i][1]=='B':
            countB+=1
        else:
            dataACD.append(data[i][0])
    else:
        if data[i][1]=='B':dataB.append(data[i][0])
dataACD.sort(reverse=True)
```

```python
for i in range(len(dataACD)):
    print(dataACD[i], dataB[i])
    if rez-dataACD[i]+dataB[i]<=S:
        rez=rez-dataACD[i]+dataB[i]
        countB+=1
print(S-rez, countB)
```

26 Во многих компьютерных системах текущее время хранится в формате «UNIX-время» – количестве секунд от начала суток 1 января 1970 года.
В одной компьютерной системе проводили исследование загруженности. Для этого в течение месяца с момента UNIX-времени 1633046400 фиксировали и заносили в базу данных моменты старта и финиша всех процессов, действовавших в этой системе.
Вам необходимо определить, какое наибольшее количество процессов выполнялось в системе одновременно на неделе, начавшейся в момент UNIX-времени 1633305600, и в течение какого суммарного времени (в секундах) выполнялось такое наибольшее количество процессов.

Входные данные
Первая строка входного файла содержит целое число N – общее количество процессов за весь период наблюдения. Каждая из следующих N строк содержит 2 целых числа: время старта и время завершения одного процесса в виде UNIX-времени. Все данные в строках входного файла отделены одним пробелом.
Если в качестве времени старта указан ноль, это означает, что процесс был активен в момент начала исследования. Если в качестве времени завершения указан ноль, это означает, что процесс не завершился к моменту окончания исследования.
При совпадающем времени считается, что все старты и завершения процессов происходят одновременно, в начале соответствующей секунды. В частности, если время старта одного процесса совпадает с временем завершения другого и других стартов и завершений в этот момент нет, то количество активных процессов в этот момент не изменяется.

В ответе запишите два целых числа: сначала максимальное количество процессов, которые выполнялись одновременно на неделе, начиная с момента UNIX-времени 1633305600, затем суммарное количество секунд, в течение которых на этой неделе выполнялось такое максимальное количество процессов.